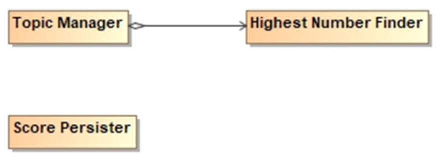
QLC-2) Find the Highest Number Extension.

An organisation delivers several topics (subjects). Students are graded against each topic. You are required to store the top score for each topic.

We've designed the application so that it comprises of three core classes:

- A class to find the highest number from an array of integers.
- A class to find the highest score for a topic.
- A class to write the topic and score to a file on the disk.



You are going to follow a TDD approach to find the highest score for a series of topics

Given the following specification

- If the input is [{"Physics", {56, 67, 45, 89}}], the result should be [{"Physics", 89}]
- If the input is [] the result should be []
- If the input is [{"Physics", {56, 67, 45, 89}}, {"Art", {87, 66, 78}], the result should be [{"Physics", 89}, {"Art", 87}]
- If the input is [{"Physics", {56, 67, 45, 89}}, {"Art", {87, 66, 78}}, {"Comp Sci", {45, 88, 97, 56}}], the result should be [{"Physics", 89}, {"Art", 87}, {"Comp Sci", 97}]

QLC-2.1) Setup and first test

1. In FindHighestNumber project create a new package that ends with `.topicmanager`
2. Create a new setup the new test class as shown here (remember you will have errors)
3.
```java
package xx.yy.zz.topicmanagerservice

public class TopicManagerTests
{
    @Test
    public void find_heighest_score_in_empty_array_return_empty_array()
    {
        // Arrange
        int[] array = {};
        TopicManager cut = new TopicManager();
        int[] expectedResult = {};

        // Act
        Topic[] result = cut.findTopicHighScores(array);

        // Assert
        Assert.That(result, Is.EqualTo(result));
    }
}
```
4. Create a new class called TopicManager in the FindHighestNumberService project

5. Clean the code up so that the class is in a package called `topicmanagerservice`
6. Lines 9, 11, and 14 give us enough information to allow us to start thinking about what we are trying to design here. Based on the requirements, we want to pass into the method `findTopicHighScores` an array of Topics and their accompanying scores. It should return the top topic score of each Topic, we will call this TopicTopScore. Each item in the array being passed into the `findTopicHighScores` will be called `TopicScores` (plural I know, but it matches the context, you may want to lose the ‘s' at the end of the class name). Let’s refactor the code to reflect what we have outlined here.
7.

```java
    public class TopicManagerTests
    {
        @Test
        public   void  find_heighest_score_in_empty_array_return_empty_array()
        {
            // Arrange
            ArrayList<TopicScores> array = new ArrayList<>();
            TopicManager cut = new TopicManager();
            ArrayList<TopicTopScore> expectedResult = new ArrayList<>();

            // Act
            ArrayList<TopicTopScore> result = cut.findTopicHighScores( array );

            // Assert
            assertEquals( expectedResult, result );

        }
    }
```

8. We are now in a good position to get VS to generate the `findTopicHighScores` with the correct signature
9.

```java
class TopicManager
{
   ArrayList<TopicTopScore> findTopicHighScores(ArrayList<TopicScores> array)
   {
      return new ArrayList<>();
   }
}
```

10. Now we need to complete the first piece of implementation code to pass the test
11. So you will also need the following pieces of code
12.

```java
package com.s2s.demos.topicmanager;

class Topic
{
   private  String topicName;
   int      score;

   public   Topic( String topicName, int score)
   {
      this.topicName = topicName;
      this.score = score;
   }

   public String getTopicName()
   {
      return topicName;
   }

```

```
    public int getScore()
    {
       return score;
    }
}
```

13.

```
public class TopicTopScore
{

}
```

14.

```
public class TopicScores
{

}
```

Second test

1. Implement this second test
2.

```
    @Test
    public void find_heighest_score_with_array_of_one_return_array_of_one()
    {
        // Arrange
        int[] scores = { 56, 67, 45, 89 };
        String topicName = "Physics";
        ArrayList<TopicScores> topicScores = new ArrayList<>();
        topicScores.add(new TopicScores(topicName, scores));

        TopicManager cut = new TopicManager();
        ArrayList<TopicTopScore> expectedResult = new ArrayList<>();
        expectedResult.add(new TopicTopScore(topicName, 89));

        // Act
        ArrayList<TopicTopScore> result = cut.findTopicHighScores(topicScores);

        assertEquals(expectedResult.get(0).getTopicName(), result.get(0).getTopicName());
        assertEquals(expectedResult.get(0).getTopScore(), result.get(0).getTopScore() );
    }
```

3. And here is the implementation class
4.

```
class TopicManager
{
   private final HighestNumberFinder highestNumberFinder = new HighestNumberFinder();

   ArrayList<TopicTopScore> findTopicHighScores(ArrayList<TopicScores> array)
   {
      ArrayList<TopicTopScore> topScores = new ArrayList<>();

      if(array.size() == 1)
      {
          TopicScores ts = array.get(0);
          int topScore = highestNumberFinder.findHighestNumber(ts.getScores());

          topScores.add(new TopicTopScore(ts.getTopicName(), topScore));
```

```java
      }
      return topScores;
   }
}
```

5. Updated TopicScores.java
6.

```java
package com.s2s.demos.topicmanager;

public class TopicScores
{
   private String topicName;
   private int[] scores;

   public TopicScores(String topicName, int[] scores)
   {
      this.topicName = topicName;
      this.scores = scores;
   }

   public String   getTopicName()
   {
      return topicName;
   }

   public   int[] getScores()
   {
      return scores;
   }
}
```

7. Updated TopicTopScores.java
8.

```java
public class TopicTopScore
{
   private  String topicName;
   private  int topScore;

   public TopicTopScore(String topicName, int score)
   {
      this.topScore = score;
      this.topicName = topicName;
   }

   public String getTopicName()
   {
      return topicName;
   }

   public int getTopScore()
   {
      return topScore;
   }
}
```

9. Looking back at TopicManager, `Line 3` represents the elephant in the code
   a. We already have tests for the `HighestNumberFinder` class, and yes class `TopicManager` requires `HighestNumberFinder`. But the coupling between these two classes is such that we cannot test `TopicManger` independently of `HighestNumberFinder`. Actually,

    this is a code smell.

    b. we can not substitute out `HighestNumberFinder`, this also is a code smell.

    c. We can remove the code smell by injecting `HighestNumberFinder` into `TopicManager`, when `TopicManager` is created.

10. Modify `TopicManager` as follows

    a. remove the creation of `HighestNumberFinder` at line 5

    b. Pass an instance of a `HighestNumberFinder` into `TopicManager` as a constructor parameter

11.
```
public class TopicManager
{
    private HighestNumberFinder highestNumberFinder;

    public  TopicManager( HighestNumberFinder hnf )
    {
        highestNumberFinder = hnf;
    }

    public TopicTopScore[] findTopicHighScores(TopicScores[] array)
    {
        if(array.Length == 1)
        {
            List<TopicTopScore> topScores = new List<TopicTopScore>();

            TopicScores ts = array[0];
            int topScore = highestNumberFinder.findHighestNumber(ts.Scores);

            topScores.Add(new TopicTopScore(ts.TopicName, topScore));

            return topScores.ToArray();
        }
        else
            return Array.Empty<TopicTopScore>();
    }
}
```

12. Refactor the tests so that they run. You should not have to change any of the test data.

Tests are a great way of identifying code smells. Highly coupled code leads to untestable code.

You want to test one class and one class only. The previous version of the `TopicManager` dragged in `HighestNumberFinder`. So inadvertently you were testing that class as well. This will become clearer as we continue to work through the `TopicManager` tests.

QLC-2.2) Working with Stubs

A Stub is part of the family of Test Doubles. They are used to ensure that tests focus on the behaviour of the CUT and not its dependents. Test environments should be controlled and predictable. Test Doubles give you that measure of stability and predictability.

A Stub method is one that returns canned results. A canned result is a predefined result. The result can be specific, a range of values, or any value. Also, the parameters into the method can be specific value, a range of values, or any value.

The `HighestNumberFinder` is designed to return an integer representing the number in a group of numbers. This can easily be stubbed out.

Begin by creating a new test

```
1      @Test
2      public void find_heighest_score_with_array_of_one_return_array_of_one_using_stub()
3      {
4          // Arrange
5          int[] scores = { 56, 67, 45, 89 };
6          String topicName = "Physics";
7          ArrayList<TopicScores> topicScores = new ArrayList<>();
8          topicScores.add(new TopicScores(topicName, scores));
9
10         // Use a stub version of HighestNumberFinder
11         com.s2s.demos.topicmanager.HighestNumberFinder hnf =
12                          new com.s2s.demos.topicmanager.HighestNumberFinder();
13         TopicManager cut = new TopicManager(hnf);
14         ArrayList<TopicTopScore> expectedResult = new ArrayList<>();
15         expectedResult.add(new TopicTopScore(topicName, 89));
16
17         // Act
18         ArrayList<TopicTopScore> result = cut.findTopicHighScores(topicScores);
19
20
21         assertEquals(expectedResult.get(0).getTopicName(), result.get(0).getTopicName());
22         assertEquals(expectedResult.get(0).getTopScore(), result.get(0).getTopScore() );
23     }
```

Here is the Stub you will use (**create it in the test folder -** it's not production code)

```
package com.s2s.demos.topicmanager;

public class HighestNumberFinder
{
    public int findHighestNumber(int[] array)
    {
        return 89;
    }
}
```

The stub is substituted in line 11.

But when you try and use this class in the tests, you should see type errors. This is because, in the implementation unit of `TopicManager`, is typed against `com.s2s.demos.findhighestnumber.fin.HighestNumberFinder`. So packages do not offer us the solution we are looking for.

We need to use interfaces. Alternatively, you could modify `TopicManager` so it has a default constructor, but you will need to think about different issues and more tests for if `TopicManager` default constructor is used.

1. Create a new interface that is part of the production code, and create it in the `com.s2s.demos.findhighestnumber` package
2.
```
package com.s2s.demos.findhighestnumber;

public interface IHighestNumberFinder
{
   int findHighestNumber(int[] values);
}
```
3. Modify the production version and test version of HighestNumberFinder, so that they both implement the interface `IHighestNumberFinder`
4. Here is the stub version

5.

```java
package com.s2s.demos.topicmanager;

import com.s2s.demos.findhighestnumber.IHighestNumberFinder;

// The STUB version
public class HighestNumberFinder implements IHighestNumberFinder
{
   @Override
    public int findHighestNumber(int[] array)
    {
        return 89;
    }
}

```

6. Here is the production version
7.

```java
package com.s2s.demos.findhighestnumber.fin;

import com.s2s.demos.findhighestnumber.IHighestNumberFinder;

public  class HighestNumberFinder implements IHighestNumberFinder
{
   @Override
    public int findHighestNumber(int[] array)
    {
        int highestSoFar = Integer.MIN_VALUE;

        for( int val : array )
        {
            if( val > highestSoFar )
                highestSoFar = val;
        }
        return highestSoFar;
    }
}
```

8. Refactor `TopicManager` so it now works with the interface `IHighestNumberFinder` and not directly with the implementation class
9.

```java
package com.s2s.demos.topicmanager;

import com.s2s.demos.findhighestnumber.IHighestNumberFinder;
import java.util.ArrayList;

class TopicManager
{
   private final IHighestNumberFinder highestNumberFinder;

   public  TopicManager()
   {
      this.highestNumberFinder = null;
   }

   public  TopicManager( IHighestNumberFinder hnf )
   {
       highestNumberFinder = hnf;
```

```
    }
    ...
```

10. You should find that the test is no longer in an error state
11. Rerun all your tests they should still be passing.

The tests have given us the confidence to refactor the code and begin to think more clearly about our implementation.

We’ve now created a controlled environment for our tests

QLC-2.3 Limitations of Stubs, complete this TopicManager requirement

Add more tests to handle the last two requirements

•If the input is [{“Physics”, {56, 67, 45, 89}}, {“Art”, {87, 66, 78}], the result should be [{“Physics”, 89}, {“Art”, 87}]

Only one of the tests passes. Why?

Sample solution

QLC-2.4 Mocks, complete the last TopicManager requirement

•If the input is [{“Physics”, {56, 67, 45, 89}}, {“Art”, {87, 66, 78}}, {“Comp Sci”, {45, 88, 97, 56}}], the result should be [{“Physics”, 89}, {“Art”, 87}, {“Comp Sci”, 97}]